Technical Report 1662
July 1994

Case Study of a Real-Time Application of an iPSC/860 Computer

P. M. Reeves

CONTENTS

INTRODUCTION

PURPOSE

The project applied an off-the-shelf Intel iPSC/860 parallel processor to an embedded application in the NRaD Hybrid Simulator. This is a hard real-time, hardware-in-the-loop simulation that is a production facility heavily used by the Navy and its contractors for the development, test, and evaluation of torpedo guidance systems. The technical benefits of this project include

- an order of magnitude increase in processing power, compared to the previous simulation configuration,
- greatly increased flexibility to meet evolving customer requirements,
- improved hardware reliability.

Achieved in parallel with these benefits was the important bottom-line benefit of substantially reduced cost in operation and maintenance.

The overall project involved many hardware and algorithm changes. This paper focuses on the key element of the upgrade—the embedding of the iPSC/860 in the simulator.

REPORT SECTIONS

The remainder of this report is divided into the following six sections:

- **Background.** Provides overviews of the simulator application, the problem description, and the project objectives.
- **Approach.** Explains principal technical and management approaches to this project.
- **Risk Reduction.** Describes some of the more significant HPC-related risks anticipated by this project, and how they were avoided, or minimized by technical and organizational factors.
- **Experience.** Discusses what NRaD has learned about problems of this type, what worked, what failed, and the benefits realized.
- **Status.** Summarizes the completed project.
- **Conclusions.** Restates principal points from the preceding sections.

BACKGROUND

This section describes the function of the NRaD hybrid simulator, its design philosophy, the need for improvement, and the specific objectives of this project.

FUNCTION

The NRaD Hybrid Simulator is a production facility used for torpedo guidance and control system Development, Test, and Evaluation (DT&E). This facility has been used by all U.S. Navy homing torpedo developments, since the early 1950s.

Modern torpedoes are extremely sophisticated. Those designed for launch from air and surface platforms must autonomously search for, detect, classify, and attack submarine targets, anywhere within a military sector consisting of many square miles of ocean area, while defeating all defensive maneuvers and countermeasures. The underwater environment, in which these torpedoes operate, is complex and difficult to measure; therefore, it is extremely difficult to observe an in-water test on the scale of a torpedo-target engagement, and impossible to control, or even measure all of the variables that influence its result. Such testing is also very expensive. Given the complexity of the torpedo and its environment, as well as the cost of in-water testing, a facility is required that can provide testing that is representative of the in-water experience, but is performed in a controlled, highly instrumented, and cost-effective manner.

A Hardware-In-The-Loop (HWIL) simulation satisfies this need. It provides a representation of the underwater environment, in which either an emulation of the torpedo guidance system or an actual torpedo guidance system can be embedded. The ability to perform realistic system testing with an emulation-in-the-loop, rather than a production guidance system, allows developers the simplicity and cost saving of testing without the requirement to field fully operational hardware and software. Typical HWIL simulator applications include: hardware and software development, debug, and test; in-water test planning; repetition of in-water tests and examination of test variations; and performance evaluation in scenarios that cannot reasonably be tested in water.

A HWIL simulator must operate in hard real-time. This is necessary, because the operational torpedo hardware does not support any other mode of operation. The torpedo's internal clock runs continuously, but its rate cannot be changed to accommodate the simulator. The simulator must therefore stimulate all of the torpedo's sensors with signals properly timed to match the real-time evolution of the scenario. If this timing is not achieved, the simulator simply will not work correctly.

DESIGN PHILOSOPHY

As already noted, HWIL simulation is a tool for torpedo system development, test, and evaluation. In this role, the NRaD simulator must provide responses that realistically represent the underwater environment over as wide a range of test scenarios as possible. Since the torpedo-under-test is complex and is typically not fully understood, it is essential that the simulator be general enough to respond realistically to unanticipated torpedo behavior. Because of this necessary generality, the requirements for model fidelity and computational capacity are high. Additionally, the simulator must provide a flexible architecture to facilitate rapid adaptation to changing customer requirements without impacting project schedules or budgets.

NEED FOR IMPROVEMENT

Over the years, the NRaD simulator has periodically been upgraded to meet new test requirements and to maintain equipment. The last major upgrade was completed in the early 1980s. At that time, the architecture was modified to employ

- a UNISYS 1100/83 mainframe as the simulator controller, and for some modest modeling computations;
- AP/120B array processors interfaced to the UNISYS machine that were used as the principal computing elements;
- Individual pieces of special-purpose hardware developed for specific functions: such as, the real-time clock, data buffering, signal filtering, and signal frequency translation.

The computational capacity of this system was approximately 225 Million Floating Point Operations Per Second (MFLOPS).

Much of the 1980 simulator software was written in FORTRAN and Concurrent Pascal; however, the high performance computing requirement was met with the use of microcode and array processor assembly language.

Since 1980, model fidelity requirements have increased as torpedo technology advanced. During this time, the simulator was modified and tuned to satisfy customer requirements. By 1990, both computational power and architectural flexibility had reached the limits of the design. Further modification to meet new user requirements was no longer practical. System operation was complex and error prone. Additionally, both reliability and operating cost of the complicated network of individual components had become problems.

PROBLEM DESCRIPTION

The specific simulation function performed by the iPSC/860 computer is the generation of digital time series that represent signals received by a multibeam torpedo active sonar. These signals must model the complex reflection and refraction of sonar pulses as they propagate through the environment, including scattering from the surface and bottom, and nonuniform volume, as well as the echoes from surface ship and subsurface submarine targets.

These acoustic effects must be modeled for many different sonar waveforms covering a wide range of spatial and frequency resolutions. The model must also account for arbitrary accelerating motion of the sonar receiver during the listen interval.

During a simulation run, many important parameters are under the autonomous control of the torpedo. Examples include: transmit pulse type; transmit time; transmit frequency; transmit beam direction and pattern; multiple receive beam directions and patterns; length of the listen interval; and the dynamic motion of the torpedo during the listen interval; thus, the simulator must be ready to respond with realistic time-series sonar signals correctly representing the sonar parameters chosen by the torpedo, as well as the instantaneous dynamic and geometric state of the simulated scenario.

In some cases, there is only a small fraction of a second between the generation of a sonar command by the torpedo and the time at which its sonar receiver begins to process data. As

already noted, this is a hard real-time problem; that is, once a sonar ping is initiated, signals on the timing expected by the torpedo must be generated by the iPSC/860 and clocked into the torpedo, or the simulation cannot function correctly.

UPGRADE OBJECTIVES

This project has four principal objectives:

- Reduced Costs. Operating and maintaining the many heterogeneous components of the simulator connected by a complex communication network was a labor intensive and expensive task. This project reduced costs by replacing many hardware components with an integrated system of new technology processors and data communications.
- Improved Hardware Reliability. Failures of individual simulator components were frequent, reducing simulator availability, and sometimes impacting customer schedules. This project replaced older components with new, more reliable, hardware technology.
- Increased Fidelity. The ability to implement new algorithms, requiring increased computational power or interprocessor communication, was severely limited, because the existing simulator components were loaded to capacity. The upgrade enabled fidelity improvements by providing an order-of-magnitude increase in peak computational power, as well as much more flexible and capable interprocessor communication.
- Improved Operator Interface. Operation of the existing simulator was complex and labor intensive. The heterogeneous environment and limited hardware capacity made such operations, as the generation of simulation setup files, run data logging, and error monitoring, both difficult and unreliable. The upgrade provided a unified environment within which these, and other, operator actions could be greatly simplified and automated for improved reliability.

APPROACH

This section describes the technical and management approaches to this project.

TECHNICAL APPROACH

This application is a naturally concurrent Multiple Instruction Multiple Data (MIMD) problem. It directly lends itself to the use of multiple processors operating in parallel. Each processor runs its own unique software, cooperating with other processors, to create the real-time simulator functionality.

Processing power, communication bandwidth between processes, communication latency, and the flexibility to expand processing power and communication bandwidth are all very important in this application. The scalable parallel MIMD architectures, now commercially available, are well-suited to this application. They provide integrated systems with the ability to simultaneously increase both computing power and interprocessor communications for a very high degree of design flexibility.

A further advantage of these machines, as opposed to the heterogeneous architecture of the 1980 simulator, is that they offer a uniform operating and development environment within which the computing elements are embedded. This greatly simplifies problem diagnosis and the implementation of system improvements.

NRaD was familiar with the Intel iPSC/860, as a result of a study assessing advanced simulation requirements and high performance computing alternatives. This project examined a number of parallel machines. It was determined that the iPSC/860, while not sufficiently powerful to satisfy long-term requirements, was very adequate to meet current needs and provided large growth potential for near-term requirements. The iPSC/860 was also one in a development sequence of increasingly powerful Intel “Touchstone” machines, so its selection offered the potential for growth to software-compatible machines able to satisfy longer-term requirements.

The 1980 simulator design was highly optimized to utilize its peak power of about 225 MFLOPS. When approaching the upgrade, the peak processing power requirement of the new hardware was conservatively set by estimating that an initial efficiency of 10% to 15% should be easily attainable; thus, the peak power of the new machine should be roughly two billion floating point operations per second (two GFLOPS). This approach promised a high probability of successfully porting the 1980 simulator capability to the new machine with little difficulty. The excess computing capacity needed for fidelity improvements could then be obtained over time as system optimization improved efficiency and created reserve computational capacity.

A 32-node iPSC/860 processor was selected to satisfy the acoustic simulation requirement. This was procured as an initial purchase of 16 nodes with later expansion to 32.

MANAGEMENT APPROACH

Strong management support of this HPC project was provided by NRaD and the Defense Advanced Research Projects Agency (DARPA). The project would not have been possible without this support.

NRaD Support

The iPSC/860 was procured by NRaD through an internal program aimed at developing corporate HPC expertise and applying HPC technology to Navy problems. Under this program, the

Center recognized that the current state-of-the-art in parallel machines requires expertise and dedication on the part of their users. The Center, therefore, purchased machines-of-interest and placed them with projects; such as, the Hybrid Simulator which was subject to the requirements that

- the machine would be made available, over existing wide-bandwidth computer networks, to other users;
- the machine would be maintained and expanded as necessary by the project;
- expertise developed by project personnel would be shared with the Center;
- the machine would be moved to another project, if the proposed project application proved unsuccessful.

In return for support of the machine, the project received top priority for its use.

This approach made the machine available to Center users, while at the same time creating a dedicated and expert support group through the project application.

DARPA Support

The NRaD high performance computing initiative was assisted by DARPA. Specifically, a joint program was established under which NRaD and DARPA cooperated on the procurement of DARPA-sponsored HPC computers. The conditions of this assistance were

- primary applications would be DARPA-approved highly challenging problems,
- NRaD would report experiences to DARPA and the machine developers,
- the machines would be made available to other users.

This assured DARPA that their HPC developments would immediately be inserted into demanding Navy applications, that operational experience would be accumulated to support further HPC development, and that the user base of these machines would expand within the DoD community.

RISK REDUCTION

There were a number of technical risks associated with the application of a state-of-the-art parallel processor to the simulator upgrade. Some of these and the corresponding risk reduction considerations are discussed below. These points provide insight on how the risk of using newly developed, parallel computing technology in an embedded applications was managed.

SCHEDULE

The Hybrid Simulator is a production facility that is depended upon by its users. Configuration changes must be introduced carefully, to maintain control of the models and continuity of project data. Insertion of the iPSC/860 upgrade into the operational simulator was limited to a window of only a few months, during which user schedules permitted the simulator configuration to change as the new system was brought on line. Meeting schedule was therefore a very important project consideration.

One key means of schedule risk reduction was the procurement of a machine with excess computational capacity. Specifically, sufficient computational power was purchased to realize the 1980 simulator power, if only 10% to 15% of the theoretical peak computing power of the new machine was achieved on the initial implementation. This minimized the likelihood that a difficult system optimization would be required. The additional capacity also facilitated the development process by allowing more programmers to work concurrently. Finally, the excess capacity was not wasted, as it could be applied to simulator fidelity improvements, as subsequent tuning improved system efficiency.

Other features that significantly reduced schedule risks were

- High experience level of the people involved. Virtually every member of the project team had years of experience with the Hybrid Simulator. They thoroughly understood the application, and therefore, recognized and avoided many subtle problems.
- Preliminary design. Much effort went into the system preliminary design, before anyone started writing or porting code. Some ideas had to be modified when the team discovered how the iPSC/860 actually worked, but basic design remained unchanged. This early effort paid off by providing a stable software development environment throughout the project.
- Requirements control. In a project such as this, there is a great temptation to “enhance” the goals as opportunities to exploit the new technology are discovered. While many enhancements of this type were allowed by project management, schedule risk versus project gain was always an important consideration. Resistance to requirements growth helped provide a stable software development environment throughout the project.

REAL-TIME OPERATION

This is a real-time application, but the iPSC/860 is not a real-time computer. There is no global clock, and there are no specifications of worst case timing. Risk was reduced again, by choosing a machine configuration with excess computational power and communication bandwidth. Additionally, the design was deliberately biased toward reduction of both internode communication and sensitivity to external even synchronization. This maximized the likelihood of

satisfying all timing constraints with minimal development effort. The lack of a global clock was overcome by providing uniform timing information to each iPSC/860 node from an external real-time clock, in combination with a synchronization algorithm that accounted for message passing delays.

MACHINE MATURITY

The iPSC/860 was new to the market when the project described in this paper began in 1990; therefore, maturity of the machine was a serious consideration. Initial problems with the hardware, operating system, and support software were all anticipated, and they did cause considerable difficulty early in the development. In time, these problems were either fixed by Intel or worked around by us.

The general issues of how quickly the system software would stabilize and how reliable would it become were more important than the individual system problems. These concerns were alleviated by the fact that the iPSC/860 is one machine model in a development series. It was preceded by the iPSC/2 that had documented operational experiences. We thus had at least some indication that the predecessor system had achieved a workable level of reliability and stability. Additionally, the Intel processor chip (the i860), used in the iPSC/860, was becoming widely used in applications other than the iPSC/860. This gave us confidence that software tools and libraries associated with the processor would likely improve with time, as a 3rd-party software market grew.

As a fallback position, the use of assembly language for critical performance areas of the software was always a possible—though very undesirable—option. The 1980 simulator made extensive use of assembler programming, and members of the development team, therefore, had expertise with this difficult mode of software development. Fortunately, this expertise was not needed for the iPSC/860 project.

HARDWARE RELIABILITY

Although the iPSC/860 was new to the market, there were already several machines in operation. Reports indicated that the inevitable early hardware problems were being eliminated. This gave us confidence that we would receive reliable hardware.

SOFTWARE DEVELOPMENT

Any project seeking to develop real-time software on a new machine must carefully consider the software development risk. In this case, risk was reduced because we already had an operating simulator, and the personnel working on the upgrade had participated in the design and development of that earlier simulator hardware and software.

Risk was also reduced by designing an extensive diagnostic and data logging facility into the new system. In addition to its use during normal simulator operations, this facility allowed for performance monitoring and assessment of individual software modules as they were being developed. Development time, integration time, and the probability of unpleasant surprises late in the development process were thereby all reduced.

Software risk was further minimized by a two-phase development plan. The first phase ported the current models to the new hardware without functional change. Once the baseline

functionality had been successfully implemented, the second phase would implement new functionality; thus, the challenge of learning to work in a new operating environment was effectively separated from the challenge of implementing new capability. For simulator customers, this approach had the important additional benefit of moving the existing simulator models to the new hardware baseline, before starting to change the models for improved fidelity.

Finally, software risk was also reduced by the use of a small, highly experienced development team. Close coordination was therefore relatively easy to maintain. Common source code control tools were used by all team members to track software changes.

HIGH LEVEL LANGUAGE

The use of a high level language implies a reliance on compilers to achieve reasonable execution efficiency. In a real-time system, both absolute execution speed and the stability of execution speed with code modifications are important. The 1980 simulator made extensive use of machine assembler coding to achieve maximum efficiency and programmer control. In this upgrade program, excess computing power was procured, so that the inefficiencies of a high level language, possibly exaggerated by an immature compiler, could be tolerated. In addition, by selecting a machine based upon processors with a wide commercial market, we were confident that compiler improvements would be realized with time.

COMMUNICATION AND I/O BANDWIDTH

The ability to move information rapidly between internal processors, as well as in and out of the machine, is critical to this application. Fortunately, the simulator problem is well-suited to a coarse-grained model utilizing powerful processors and comparatively little interprocessor communication. This model fits the iPSC/860 well. Bandwidth out of the machine is minimized by representing the simulated sonar signals as complex baseband. (Conversion of these digital time series to analog signals at the torpedo sonar frequency is handled by external processors.)

The operator interface to the iPSC/860 is handled by a direct Ethernet connection to one node of the hypercube. This bypasses the slow Intel user interface, an i386-based micro computer.

EXPERIENCE

This section briefly describes some of the key development experiences and issues in this project.

RECEIPT AND SETUP

This was handled by Intel. The process required essentially one day. No problems were encountered.

MANUFACTURER SUPPORT

We received good cooperation from Intel engineers during the project phases, when interfacing through electronic mail, telephone, and meetings.

LEARNING CURVE

The length of time required for the development team to learn the iPSC/860 system basic was relatively short. Simulator software development was underway approximately one month after delivery of the new machine, even though the majority of the developers had no significant prior experience with the C language used in this project.

SOFTWARE DEVELOPMENT ENVIRONMENT

Code development and configuration control were performed on a network of workstations. Effective commercial instrumentation and performance evaluation tools were lacking throughout this project. It was the responsibility of project personnel to develop their own techniques for instrumenting the system and assessing performance.

Software code compilation was a potential bottleneck. Intel provides an i386-based host computer for the iPSC/860 known as the System Resource Manager (SRM). The SRM, which is normally used for code compilation, is poorly suited to a project of this magnitude and is too slow for practical use. Cross-compilation on a Sun workstation allowed the development team to avoid the SRM, and it worked well.

Finally, the iPSC/860 services multiple users by allocating a distinct subset of the available compute nodes to each. This limited the number of people who could have simultaneously used the machine. It could have been a project bottleneck; however, during development the iPSC/860 was shared by members of the development group with minimal difficulty.

PROCESS MAPPING

The port of the 1980 simulation algorithms to the new processor was made considerably less complex by a good match between the coarse granularity of the problem and the design of the iPSC/860. Individual algorithms did not have to be processed in parallel across multiple processors, because each i860 processor has very large computing power and substantial local memory. A total of 16 compute processors and four I/O nodes are currently being used.

SOFTWARE TRANSLATION

High level languages used in the 1980 simulator were FORTRAN and Pascal, whose code was also used as an executable specification of the array processor assembler code; thus the

principal porting effort was to move FORTRAN and Pascal onto the iPSC/860, where the available languages are FORTRAN and C. The FORTRAN code was essentially ported directly. The Pascal code was translated automatically to C, followed by manual review and rewriting as necessary. Process control and communication code was entirely rewritten to match the iPSC/860 architecture.

SOFTWARE COMPILATION

Initial performance of the C compiler was poor, but the situation improved with time. Performance of the current C compiler (from the Portland Group) is now sufficient to support real-time operation and continued improvement is expected.

Consistent retention of the ANSI C standard is a possible concern as compilers are upgraded, or new compilers are developed. We found that use of this standard, combined with development tools to verify conformance, resulted in significant debugging and integration time savings.

Since we wanted to interface special-purpose hardware to the iPSC/860, it was necessary to reprogram the standard Intel I/O nodes. Separate compilers, debuggers, and libraries were required, and Intel operational support was less effective than for i860 nodes. This caused some inconvenience, but not great difficulty.

PROCESSOR ASSIGNMENT

The standard iPSC/860 system assigns a group of compute nodes to the user in a “cube,” the size of which is an integral power of two. For the simulator application, it is necessary to link specific processes to specific I/O nodes; thus, software was developed that allows the user to assign compute nodes by absolute address, and it is not limited to an integer power of two. This provides the ability to assign processes to specific locations within the hypercube, and it might be used to optimize interprocess communications, although this is not required at present.

PROCESS SYNCHRONIZATION

This important aspect of the real-time system was, to some extent, provided naturally by the message passing architecture of the machine; however, the lack of a global system clock is a serious deficiency for real-time applications. As noted above, we overcame this problem by implementing a scheme that uses an external clock and passes timing information with messages; however, much of this could have been avoided by an internal global clock.

PROCESS RAPID RESTART

This is a special requirement of a real-time system. It refers to the need to terminate a process when its result becomes irrelevant. This condition arises in the simulator application when, for example, the embedded system-under-test changes its operating mode. Old processes must immediately be killed, and new processes must be started to service the new mode. Intel did not anticipate the need for system support of such a capability in the iPSC/860, and this was a difficult problem to solve. The solution is now implemented by a scheme using flag setting and polling. This is not an elegant solution, but it is serviceable.

DEBUGGING AND PERFORMANCE MONITORING

These are critically important for any parallel processor or real-time application, and, therefore, doubly important for this combined application. Tools provided by the manufacturer may be adequate for more standard parallel processing applications, but were not generally useful to this project, due to the real-time requirement. This problem was compounded by the lack of a global clock. A limited performance monitoring capability was implemented by using hardware ports on the iPSC/860 compute nodes for precision timing measurements. Primary reliance is upon a user-developed, message-passing system that also supports operator control functions and run data logging.

STATUS

The iPSC/860 processor was received at NRaD near the end of September 1990, and an initial closed-loop operating capability was achieved at the beginning of October 1991—one year and two days later. Signaling the start of production operations with the new system, full operating capability was reached in January 1992. Estimates of the computing capacity needed for the initial implementation proved to be conservative. The phase-one simulator configuration required only 16 compute nodes.

The second project phase, implementation of fidelity improvements, was completed at the end of September 1992. As the system was optimized during the fidelity upgrade process, it was possible to implement new and more complex models, while at the same time significantly increasing the update rates of the models. The phase-two simulator configuration was implemented in the same 16 nodes required by the phase-one configuration.

Both phases of the project were accomplished on schedule and within budget. Compared to the 1980 simulator, performance is substantially improved and operating costs are much lower. Cost saving exceeded the iPSC/860 procurement cost in October 1992 (nine months of operation). It is expected that the savings will exceed total procurement and software development costs by October 1993 (21 months of operation).

Other NRaD projects are timesharing the use of the iPSC/860. The machine is being applied to undersea, acoustic surveillance research, and to investigations of instrumentation tools for real-time parallel computing.

CONCLUSIONS

This paper has described the successful application of an off-the-shelf iPSC/860 parallel processor to a challenging real-time embedded application. Many technical and organizational factors entered into the formula for success.

Some of the principal points from the foregoing are summarized as follows:

- This project was a joint NRaD/DARPA cooperation to demonstrate the insertion of HPC technology into challenging Navy problems.
- The project has significantly reduced simulator operating costs, while at the same time improving system reliability, upgrading the operator interface, and providing expanded computational resources to support fidelity improvements. Estimated cost savings will repay both hardware and software costs of the project in 21 months of operation.
- Although there were many risks in a project such as this, the risks were reduced by a combination of several of the following conservative policies.
  - A two-phase plan was used to separate the risk of learning a new machine from the risk of implementing software improvements.
  - The first-phase design was sized to require only 10% to 15% of the peak computational power.
  - A state-of-the-art machine was selected that was based on a widely used commercial processor chip that was part of an already successful development series.
- Favorable experiences. The several favorable experiences from this project were the short learning curve required by the iPSC/860, good manufacturer support, and the good match of machine characteristics to the granularity of our problem.
- Unfavorable experiences. Two unfavorable experiences during the project were (1) little support software, especially the early C compiler, and (2) the lack of a global clock.
- Status. The project was completed in September 1992. All work was completed on schedule and within budget. All program performance goals were met.